

[Document number: P1451.5-Prop1\\_V1](#)

IEEE P1451.5  
Wireless Sensor Interface Working Group

Security Proposal  
Revision 1.1~~0~~

[Updated 4/8/03](#)

Prepared by:

R. K. Coleman

**3e Technologies International, Inc.**  
700 King Farm Boulevard, Suite 600  
Rockville, MD 20850

### **Acknowledgements**

This security proposal leverages:

- Work that has been presented in IEEE Draft P802.15.4/D18 dated February 2003.
- Concepts from Bruce Schneier's *Applied Cryptography* 2<sup>nd</sup> edition.
- 3e Technologies security solutions.

### **Disclaimer**

This security proposal is intended to be integrated into the overarching IEEE P1451.5 standard, and is not intended to remain as a stand-alone document.

# Table of Contents

1.0 Security .....	<del>65</del>
1.1 Security services .....	<del>65</del>
1.1.1 Access control .....	<del>65</del>
1.1.2 Data encryption .....	<del>65</del>
1.1.3 Frame integrity .....	<del>65</del>
1.1.4 Sequential freshness .....	<del>76</del>
1.2 Security modes .....	<del>76</del>
1.2.1 Unsecured mode .....	<del>76</del>
1.2.2 ACL mode .....	<del>76</del>
1.2.3 Secured mode .....	<del>76</del>
1.3 Frame security .....	<del>87</del>
1.3.1 ACL entries .....	<del>109</del>
1.3.2 Functional description of unsecured mode .....	<del>109</del>
1.3.3 Functional description of ACL mode .....	<del>109</del>
1.3.4 Functional description of secured mode .....	<del>1140</del>
1.3.4.1 Processing outgoing frames in secured mode .....	<del>1244</del>
1.3.4.2 Processing incoming frames in secured mode .....	<del>1342</del>
1.4 Security suite specifications .....	<del>1514</del>
1.4.1 Security suite building blocks .....	<del>1514</del>
1.4.1.1 Bit ordering .....	<del>1514</del>
1.4.1.2 Concatenation .....	<del>1615</del>
1.4.1.3 Integer encoding and counter incrementing .....	<del>1615</del>
1.4.1.4 ECB encryption .....	<del>1645</del>
1.4.1.5 CTR encryption .....	<del>1746</del>
1.4.1.6 CBC encryption .....	<del>1746</del>
1.4.1.7 CBC-MAC authentication .....	<del>1746</del>
1.4.1.8 CTR + CBC-MAC (CCM) combined encryption and authentication .....	<del>1847</del>
1.4.1.9 AES encryption .....	<del>1847</del>
1.4.1.10 PIB security material .....	<del>1948</del>
1.4.2 AES ECB security suite .....	<del>2049</del>
1.4.2.1 Data formats .....	<del>2049</del>
1.4.2.2 Security parameters .....	<del>2120</del>
1.4.2.3 Security operations .....	<del>2120</del>
1.4.3 AES CTR security suite .....	<del>2322</del>
1.4.3.1 Data formats .....	<del>2322</del>
1.4.3.2 Security parameters .....	<del>2423</del>
1.4.3.3 Security operations .....	<del>2524</del>
1.4.4 AES CBC security suite .....	<del>2625</del>
1.4.4.1 Data formats .....	<del>2625</del>
1.4.4.2 Security parameters .....	<del>2827</del>
1.4.4.3 Security operations .....	<del>2827</del>
1.4.5 AES CCM security suite .....	<del>2928</del>

1.4.5.1 Data formats .....	<a href="#">29</a> <del>28</del>
1.4.5.2 Security parameters .....	<a href="#">30</a> <del>29</del>
1.4.5.3 Security operations .....	<a href="#">31</a> <del>30</del>
1.4.6 AES CBC-MAC security suite.....	<a href="#">32</a> <del>31</del>
1.4.6.1 Data formats .....	<a href="#">32</a> <del>31</del>
1.4.6.2 Security parameters .....	<a href="#">33</a> <del>32</del>
1.4.6.3 Security operations .....	<a href="#">33</a> <del>32</del>
<b>Annex A: Security implementation.....</b>	<b><a href="#">35</a><del>34</del></b>
A.1 AES ECB Mode Operation .....	<a href="#">35</a> <del>34</del>
A.2 AES CTR Mode Operation .....	<a href="#">36</a> <del>35</del>
A.3 AES CBC Mode Operation .....	<a href="#">37</a> <del>36</del>
A.4 AES CCM Mode Operation .....	<a href="#">39</a> <del>38</del>
A.4.1 Inputs .....	<a href="#">39</a> <del>38</del>
A.4.2 Authentication .....	<a href="#">40</a> <del>39</del>
A.4.3 Encryption .....	<a href="#">41</a> <del>40</del>
A.4.4 Output.....	<a href="#">42</a> <del>41</del>
A.4.5 Decryption.....	<a href="#">42</a> <del>41</del>
A.4.6 Restrictions.....	<a href="#">43</a> <del>42</del>
A.4.7 List of symbols .....	<a href="#">43</a> <del>42</del>
A.5 AES CBC-MAC Mode Operation.....	<a href="#">44</a> <del>43</del>

## **1.0 Security**

This standard is intended to accommodate a broad range of applications which impose significant constraints on requiring a baseline security implementation in the MAC sublayer; however, it is a clear requirement that basic security services must be provided which ensure interoperability and a reasonable level of protection among all devices implementing this standard. This baseline includes the ability to maintain an access control list and use symmetric cryptography to protect transmitted frames. The ability to perform this security does not imply, however, that security shall be used at any given time by any given device. The higher layers determine when security is to be used at the MAC sublayer and provide all keying material necessary to provide the security services. Key management, device authentication, and freshness protection may be provided by the higher layers, but are outside the scope of this document. An introduction to several critical security functions is presented below, followed by a description of the security suite specifications that are designed to supply these critical security functions and services.

### **1.1 Security services**

The security mechanisms in this standard are symmetric-key based using keys provided by higher layer processes. The management and establishment of these keys is the responsibility of the implementer. The security provided by these mechanisms assumes the keys are generated, transmitted, and stored in a secure manner.

#### **1.1.1 Access control**

Access control is a security service that provides the ability for a device to select the other devices with which it is willing to communicate. In this standard, if the access control service is provided, a device shall maintain a list of devices in its access control list (ACL) from which it expects to receive frames.

#### **1.1.2 Data encryption**

In this standard data encryption is a security service that uses a symmetric cipher to protect data from being read by parties without the cryptographic key. Data may be encrypted using a key shared by a group of devices (typically stored as the default key) or using a key shared between two peers (typically stored in an individual ACL entry). In this standard, data encryption may be provided on beacon payloads, command payloads, and data payloads (not on Acknowledgment frames).

#### **1.1.3 Frame integrity**

In this standard frame integrity is a security service that uses a message integrity code (MIC) to protect data from being modified by parties without the cryptographic key. It further provides assurance that data came from a party with the cryptographic key. In this standard, integrity may be provided on data frames, beacon frames, and command frames (not on Acknowledgment frames). The key used to provide frame integrity may be shared by a group of devices (typically stored as the default key) or by two peers (typically stored in an individual ACL entry).

#### **1.1.4 Sequential freshness**

Sequential freshness is a security service that uses an ordered sequence of inputs to reject frames that have been replayed. When a frame is received, the freshness value is compared with the last known freshness value. If the freshness value is newer than the last known value, the check has passed and the freshness value is updated to the new value. If the freshness value is not newer than the last known freshness value, the check has failed. This service provides evidence that the received data is newer than the last data received by that device, but it does not provide a strict sense of time.

### **1.2 Security modes**

Depending on the mode in which the device is operating and the security suite selected, the MAC sublayer may provide different security services.

#### **1.2.1 Unsecured mode**

Since security is not used for unsecured mode, no security services are provided by devices operating in unsecured mode.

#### **1.2.2 ACL mode**

Devices operating in ACL mode provide limited security services for communications with other devices. While in ACL mode, the higher layer may choose to reject frames based on whether the MAC sublayer indicates that a frame is purported to originate from a specific device. Since cryptographic protection is not provided in the MAC sublayer in this mode, the higher layer should implement other mechanisms to ensure the identity of the sending device. In ACL mode, the service that is provided is access control.

#### **1.2.3 Secured mode**

Devices operating in secured mode may provide any of the security services defined above. The specific security services are dependent on the security suite in use and these

services are specified by the security suite itself. Services that may be provided while in secured mode include:

- Access control
- Data encryption
- Frame integrity
- Sequential freshness

### 1.3 Frame security

The MAC sublayer is responsible for providing security services on specified incoming and outgoing frames when requested to do so by the higher layers. This standard supports the following security services:

- Access control
- Data encryption
- Frame integrity
- Sequential freshness

The protocol also provides the following security modes:

- Unsecured mode
- ACL mode
- Secured mode

The information determining how to provide the security is found in the MAC PIB. Refer to the two tables immediately below for details.

<b>Attribute</b>	<b>Identifier</b>	<b>Type</b>	<b>Range</b>	<b>Description</b>	<b>Default</b>
<i>macACLEntry-DescriptorSet</i>	0x70	Set of ACLDescriptor values. See Table 2	Variable	A set of ACL entries, each containing address information, security suite information and security material to be used to protect frames between the MAC sublayer and the specified device	Null set
<i>macACLEntry-DescriptorSet-Size</i>	0x71	Integer	0x00-0xff	The number of entries in the ACL descriptor set	0x00
<i>macDefaultSecurity</i>	0x72	Boolean	TRUE or FALSE	This indicates whether the device is able to transmit secure frames to or accept secure frames from devices that are not explicitly listed in the ACL. It is also used to communicate with multiple devices at once. A value of	FALSE

				TRUE indicates that such transmissions are permitted	
<i>macACLDefaultSecurityMaterialLength</i>	0x73	Integer	0x00-0x2a	The number of octets contained in <i>ACLSecurityMaterial</i>	0x1a
<i>macDefaultSecurityMaterial</i>	0x74	octet string	Variable	The specific security material to be used to protect frames between the MAC sublayer and devices not in the ACL	Empty string
<i>macDefaultSecuritySuite</i>	0x75	Integer	0x00-0x05	The unique identifier of the security suite to be used to protect communications between the MAC and devices not in the ACL as specified in Table 4.	0x00
<i>macSecurityMode</i>	0x76	Integer	0x00-0x02	The identifier of the security mode in use. 0x00 = Unsecured mode 0x01 = ACL mode 0x02 = Secured mode	0x00

**Table 1 – MAC PIB security attributes**

<b>Name</b>	<b>Type</b>	<b>Range</b>	<b>Description</b>	<b>Default</b>
ACLExtendedAddress	IEEE Address	Any valid 64-bit device address	The 64-bit extended IEEE address of the device in this ACL entry	Device specific
ACLShortAddress	Integer	0x0000-0xffff	The 16-bit short address of the device in this ACL entry. A value of 0xffff indicates that the device is only using its 64-bit extended address. A value of 0xffff indicates that this value is unknown	0xffff
ACLPANId	Integer	0x0000-0xffff	The 16-bit PAN identifier of the device in this ACL entry	Device specific
ACLSecurityMaterialLength	Integer	0-42	The number of octets contained in <i>ACLSecurityMaterial</i>	26
ACLSecurityMaterial	octet string	Variable	The specific keying material to be used to protect frames between the MAC sublayer and the device indicated by the associated ACLExtendedAddress	Empty string
ACLSecuritySuite	Integer	0x00-0x05	The unique identifier of the security suite to be used to protect communications between the MAC sublayer and the device indicated by the associated ACLExtendedAddress as specified in Table 4.	0x00

**Table 2 – Elements of ACL entry descriptor**

### 1.3.1 ACL entries

The MAC PIB security attributes contain a single default ACL entry and a number of additional ACL entries. The default ACL entry is known by every device in the PAN and is used in situations in which the device needs to communicate with a second device or with multiple devices that it may not know individually. Individual ACL entries are used in situations in which the device shares a key with a specific known device.

The default ACL entry consists of *macDefaultSecurity*, which indicates if security is in use for devices not in the ACL, *macDefaultSecuritySuite*, which indicates the default security suite to use for frames to be sent or received from devices not in the ACL, and *macDefaultSecurityMaterial*, which indicates the keying material to use in secure communications involving frames to be sent or received from devices not in the ACL. If *macDefaultSecurity* is set to FALSE, *macDefaultSecuritySuite* and *macDefaultSecurityMaterial* shall not be used.

The additional ACL entries are contained in *macACLEntryDescriptorSet*. Each ACL entry corresponds to a trusted device and consists of its PAN identifier, its 64-bit extended address, its short address (or 0xffff if this is not known) and its security suite and related keying material.

### 1.3.2 Functional description of unsecured mode

Unsecured mode is the default security mode for the MAC sublayer and provides no MAC sublayer security. A device operating in unsecured mode shall not utilize the ACL entries and shall not perform any security related operations on incoming frames. When a device receives a frame while in this mode, the MAC sublayer shall perform its filtering operations on the incoming frame before checking the **security enabled** subfield. If the security enabled subfield in the frame is set to 1 and the device is not performing an active or passive scan, the MAC sublayer shall pass the frame to the next higher layer. If the device is performing an active or passive scan, the device shall accept beacon frames with the security enabled subfield set to 1. If the MAC sublayer receives a data frame with the security enabled subfield set to 0, it shall pass the frame to the next higher layer.

### 1.3.3 Functional description of ACL mode

ACL mode provides a mechanism for the MAC sublayer to indicate to the higher layer if a received frame purportedly originated from a device in the ACL. A device operating in ACL mode shall not make any modifications to the MAC frames or perform any cryptographic operations on the frames. As a result ACL mode provides only a means for the device to filter received frames according to the source address in the frame, but not a means to securely determine the originator of the frame. When a device receives a frame while in this mode, the MAC sublayer shall perform its filtering operations on the incoming frame before checking the security enabled subfield. If the security enabled

subfield in the frame is set to 1 and the device is not performing an active or passive scan, the MAC sublayer shall pass the frame to the next higher layer. If the device is performing an active or passive scan, the device shall accept beacon frames with the security enabled subfield set to 1 and set the ACLEntry field to the *macSecurityMode* parameter value from the ACL entry associated with the sender of the data frame, if present. If the sender of the data frame was not found in the ACL, the value 0x08 shall be used in the ACLEntry field. If the MAC sublayer receives a data frame with the security enabled subfield set to 0, it shall pass the frame to the next higher layer. This is achieved by issuing a DATA.indication primitive with the SecurityUse field set to FALSE and the ACLEntry field set to the *macSecurityMode* parameter value from the ACL entry associated with the sender of the data frame, if present. If the sender of the data was not found in the ACL, the ACLEntry field shall be set to 0x08. The device shall determine if a device is in the ACL by searching *macACLEntryDescriptorSet* for an individual ACL entry with a value of ACLPANId that matches the received PAN identifier and a value of ACLExtendedAddress or ACLShortAddress that matches the received source address. If no source address is present in the frame, the ACLEntry field shall be set to 0x08.

### 1.3.4 Functional description of secured mode

Secured mode provides a mechanism for the MAC sublayer to both use the ACL functionality and provide cryptographic protection on incoming and outgoing frames. While in this mode, if the MAC sublayer receives an incoming frame or a request from the higher layers to transmit a frame, the MAC sublayer shall process the frame as outlined below. The subclauses refer to the partial table directly below.

Name	Type	Valid range	Description
...	...	...	...
SecurityUse	Boolean	TRUE or FALSE	An indication of whether the received beacon frame is using security. This value is set to TRUE if the security enable subfield was set to 1 or FALSE if the security enabled subfield was set to 0
ACLEntry	Integer	0x00-0x08	The <i>macSecurityMode</i> parameter value from the ACL entry associated with the sender of the data frame. This value is set to 0x08 if the sender of the data frame was not found in the ACL
SecurityFailure	Boolean	TRUE or FALSE	The parameter is set to TRUE if there was an error in the security processing of the frame. Otherwise, the parameter is set to FALSE.

**Table 3 – Security elements of PAN descriptor**

### 1.3.4.1 Processing outgoing frames in secured mode

While in secured mode, if the MLME receives a message from a higher layer to prepare a secure frame for transmission (i.e. the **security enabled** bit in the TxOptions is set to 1), the MLME shall scan the entries in the ACL for the correct entry to use. The MLME shall first search through *macACLEntryDescriptorSet* to find an entry in which the ACLPANId and the ACLExtendedAddress or ACLShortAddress fields match the destination address information of the frame to be created. If a match is found, the MLME shall select the security suite from the associated ACLSecuritySuite field and the security material from the associated ACLSecurityMaterial field for use on the outgoing frame.

If the MLME is unable to locate an ACLPANId and an ACLExtendedAddress or ACLShortAddress that matches the destination address information of the frame to be created, the MLME shall examine *macDefaultSecurity*. If *macDefaultSecurity* is equal to TRUE, the MLME shall select the security suite indicated in *macDefaultSecuritySuite* and the security material from *macDefaultSecurityMaterial* for use on the outgoing frame.

If the MLME is unable to locate an ACLPANId and an ACLExtendedAddress or ACLShortAddress that matches the destination address information of the frame to be created and *macDefaultSecurity* is equal to FALSE, the MLME shall inform the next higher layer.

After the MLME obtains the appropriate security suite and security material from the ACL, the MLME shall first set the security enabled subfield in the frame control field to 1 before performing the cryptographic operations on the frame.

If the security suite specifies the use of encryption, the encryption operation shall be applied only to the data in the payload field within the MAC payload, i.e. the beacon payload field, command payload field, or data payload field, depending on the frame type. The remaining fields shall be left unencrypted. If a frame does not contain a payload field, encryption shall not be used. The result of the encryption operation shall be inserted into the payload field of the frame in the place of the original data.

If the security suite specifies the use of an integrity code, the integrity code shall be applied to the MAC header concatenated with the MAC payload. The result of the integrity code computation shall be placed in the payload field within the MAC payload of the frame in addition to any other data in the payload field. If the payload field does not contain any data, then it shall contain only the integrity code. Integrity codes shall not be used for acknowledgement frames.

The ordering and exact manner of performing the encryption and integrity operations and the placement of the resulting encrypted data or integrity code within the beacon payload, command payload, or data payload field is defined by the selected security suite.

If any of the security operations fail, the MLME shall not transmit the requested frame and inform the next higher layer.

If the security operations have been successfully performed and the payload field within the MAC payload has been modified appropriately, the device shall compute the FCS over the modified frame.

#### **1.3.4.2 Processing incoming frames in secured mode**

Any incoming frame may be protected by security. If the MLME receives a frame while in secured mode, it shall perform its filtering operations on the incoming frame before checking the security enabled subfield to determine if security is used for that frame.

If the security enabled subfield of the frame control field is set to 0 and the incoming frame is an association request command frame, the MLME of the coordinator shall pass the frame information to the next higher layer. If the security enabled subfield of the frame control field is set to 0 and the incoming frame is a beacon request command frame, the coordinator shall process the frame. If the security enabled subfield of the frame control field is set to 0 and the device is performing an active or passive scan, the device shall accept beacon frames and set the SecurityUse and SecurityFailure fields of the PAN descriptor corresponding to each received beacon to FALSE and TRUE, respectively, and the ACLEntry field to the *macSecurityMode* parameter value from the ACL entry associated with the sender of the data frame, if present. If the sender of the data was not found in the ACL, the value 0x08 shall be used in the ACLEntry field (see table 3). Otherwise, if the security enabled subfield of the frame control field is set to 0, the device shall pass the frame to the next higher layer. This is achieved by issuing a DATA.indication primitive with the SecurityUse field set to FALSE and the ACLEntry field set to the *macSecurityMode* parameter value from the ACL entry associated with the sender of the data frame, if present. If the sender of the data was not found in the ACL, the ACLEntry field shall be set to 0x08.

If the security enabled subfield in the frame control field is set to 1, the MLME shall scan the entries in the MAC PIB security attributes for the correct entry to use. The MLME shall first search through *macACLEntryDescriptorSet* to find an entry in which the ACLPANId, ACLExtendedAddress and ACLShortAddress fields match the source address information of the frame received. If a match is found, the MLME shall select the security suite from the associated ACLSecuritySuite field and the security material from the associated ACLSecurityMaterial field for use on the incoming frame.

If the MLME is unable to locate an ACLPANId and an ACLExtendedAddress or ACLShortAddress that matches the source address information of the received frame, the MLME shall examine *macDefaultSecurity*. If *macDefaultSecurity* is equal to TRUE, the MLME shall select the security suite from *macDefaultSecuritySuite* and the security material from *macDefaultSecurityMaterial* for use on the incoming frame.

If the MLME is unable to locate an ACLPANId and an ACLExtendedAddress or ACLShortAddress that matches the source address information of the received frame and *macDefaultSecurity* is equal to FALSE, and the device is not performing an active or passive scan, the MLME shall pass the frame to the next higher layer. This is achieved by issuing a DATA.indication primitive with the SecurityUse field set to FALSE and the ACLEntry field set to 0x08. If the device is performing an active or passive scan, the device shall accept secure beacon frames for which the corresponding security material cannot be found and set the SecurityUse, ACLEntry, and SecurityFailure fields of the PAN descriptor corresponding to that beacon to TRUE, 0x08, and TRUE, respectively (see Table 3).

After the MLME obtains the appropriate security suite and security material values from the ACL, the MAC sublayer shall apply the security operations, defined by these values, to the frame.

If the security suite specifies the use of encryption, the decryption operation shall be applied only to the data in the payload field within the MAC payload, i.e. the beacon payload field, command payload field, or data payload field, depending on the frame type. The remaining fields shall be left unencrypted. If a frame does not contain a payload field, decryption shall not be used. The result of the decryption operation shall be inserted into the payload field of the frame in the place of the original encrypted data.

If the security suite specifies the use of an integrity code, the integrity code shall be checked by first removing the integrity code and any other security suite specific data (e.g. the frame counter and key sequence counter) from the payload field within the MAC payload and then verifying the integrity code on the MAC header concatenated with the MAC payload.

The ordering and exact manner of performing the decryption and integrity operations and the location of the security data within the payload field is defined by the security suite in use.

If at least one of the security operations fail and the device is not performing an active or passive scan, the MLME shall discard the frame and inform the next higher layer. If the device is performing an active or passive scan, the device shall accept beacon frames that cause a security operation failure and set the SecurityUse and SecurityFailure fields of the PAN descriptor to TRUE and set the ACLEntry field to TRUE if the key used in the operations was found in *macACLEntryDescriptorSet* or FALSE if the key used in the operations was found in *macDefaultSecurityMaterial* (see Tables 3, 1).

If the security operations have been successfully performed and the payload field within the MAC payload has been modified appropriately, the device shall then continue to process the frame. In the indication of the frame to the higher layer, the MAC shall set the SecurityUse field to TRUE and set the ACLEntry field to TRUE if the key used in the operations was found in *macACLEntryDescriptorSet* or FALSE if the key used in the operations was found in *macDefaultSecurityMaterial*.

## 1.4 Security suite specifications

Security suites may be used when a device is operating in secured mode. A security suite consists of a set of operations to perform on MAC frames that provide security services. The security suite name indicates the symmetric cryptography algorithm and mode. In this standard, all security suite block sizes, key lengths, and integrity code lengths are 128-bits. For all security suites in this standard, the cryptographic algorithm used shall be AES. Each device that implements security shall support the AES ECB security suite and zero or more additional security suites. Each security suite is specified by a one-octet identifier as shown in Table 4 below. An identifier of 0x00 indicates that secured mode is not to be used.

Identifier	Security suite name	Security Services			
		Access control	Data encryption	Frame integrity	Sequential freshness (optional)
0x00	None				
0x01	AES ECB	X	X		X
0x02	AES CTR	X	X		X
0x03	AES CBC	X	X		X
0x04	AES CCM	X	X	X	X
0x05	AES CBC-MAC	X		X	

Table 4 – Security suite list

### 1.4.1 Security suite building blocks

The following definitions and methods are defined for use in the security suites specified in this standard.

#### 1.4.1.1 Bit ordering

For security operations in this standard, a bit is defined to be an element of the set {0, 1}. An octet (also called a byte) is defined to be a bit string of length 8 arranged in the order in which it would be transmitted, where bit 7 is the first bit in the octet and bit 0 is the last bit in the octet. An octet string (also called a byte string) is an array of octets arranged in order with the most significant octet first. The terms first and last and leftmost and rightmost are used to distinguish the ends of octet strings (first and leftmost are equivalent; last and rightmost are equivalent). Within an octet, the terms first and last, leftmost and rightmost and high-order and low-order are used for the order of the bits (first, leftmost and high-order are equivalent; last, rightmost and low-order are equivalent).

Note that the first bit in an octet string is indexed as bit 7 of octet 0 and represents the high-order bit of the first octet. The sixteenth bit in an octet string is indexed as bit 0 of octet 1 and represents the low-order bit of the second octet.

### 1.4.1.2 Concatenation

In this standard, concatenation of two octet strings  $a$  and  $b$  of length  $m$  and  $n$  respectively, denoted  $a\&b$ , consists of the octet string of length  $m+n$  with the leftmost  $m$  octets equal to  $a$  and the rightmost  $n$  octets equal to  $b$ .

### 1.4.1.3 Integer encoding and counter incrementing

Unless otherwise stated, for security operations in this standard, when an integer is represented as an octet string, the first octet (octet 0) corresponds to the most significant octet and the first bit (bit 7) within the first octet corresponds to the most significant bit. An octet string  $A$  of length  $n$ , is written as a bit string  $A = a_{0,7}a_{0,6}a_{0,5}\dots a_{n-1,1}a_{n-1,0}$ . An octet string or bit string is converted to an integer  $I$  by assigning each bit  $a_{i,j}$  the value  $a_{i,j} * 2^{8i+j}$  and setting  $I$  to be the sum of all of the values.

As an example of integer encoding, consider the integer 11146, which corresponds to the octet string composed of the two octets 0x2b8a. This integer can be represented as the bit string 0010 1011 1000 1010. In this case, the most significant octet (octet 0) is 0x2b and the least significant octet (octet 1) is 0x8a. The least significant bit (bit 0) of octet 0 has the value 1, as does the most significant bit (bit 7) of octet 1. The integer 1 can be represented as a four octet string by 0x00 00 00 01 and as a bit string by 0000 0000 0000 0000 0000 0000 0000 0001.

The counter incrementing operation in this standard takes as input an integer that is encoded as an octet string of length  $n$ . The integer shall be incremented (increased by 1) when the counter incrementing operation is invoked. If the incremented integer is less than  $2^{8n}$ , the operation shall return the octet string encoding of the new integer. Otherwise the operation shall set the counter to  $2^{8n}-1$  and return an error (the counter incrementing operation would have caused the integer value to roll over).

### 1.4.1.4 ECB encryption

The AES electronic codebook mode (ECB) symmetric encryption algorithm used in this standard provides data encryption with the convenience of being the easiest and fastest block cipher mode of AES to implement, and encryption processing is parallelizable. Encryption consists of operating on a 128-bit block of plaintext, using a 128-bit key, to produce a 128-bit block of ciphertext. Decryption consists of operating on a 128-bit block of ciphertext, using the 128-bit key, to produce a 128-bit block of plaintext.

All of the above operations shall be performed as specified in Annex A, which defines ECB encryption in the general sense. A security suite implementing ECB encryption will specify any parameters left unspecified in Annex A according to the requirements of that particular security suite.

#### **1.4.1.5 CTR encryption**

The AES counter mode (CTR) symmetric encryption algorithm used in this standard consists of the generation of a key stream using a block cipher in counter mode, with a given key and nonce, and performing an exclusive-OR of the key stream with the plaintext and integrity code. A nonce is either a time stamp, a counter, or a special marker intended to prevent unauthorized message replay. The decryption operation consists of the generation of the key stream and the exclusive-OR of the key stream with the ciphertext to obtain the plaintext.

All of the above operations shall be performed as specified in Annex A, which defines CTR encryption in the general sense. A security suite implementing CTR encryption will specify any parameters left unspecified in Annex A according to the requirements of that particular security suite.

#### **1.4.1.6 CBC encryption**

The AES cipher block chaining mode (CBC) symmetric encryption algorithm used in this standard includes a feedback mechanism added to the AES block cipher. Encryption consists of exclusive-ORing the 128-bit block of current plaintext with the previous ciphertext block and then encrypting the 128-bit exclusive-OR output, using a 128-bit key. Decryption consists of operating on the 128-bit block of current ciphertext, using the 128-bit key, then exclusive-ORing the current cipher output with the previous ciphertext block to produce a 128-bit block of plaintext.

All of the above operations shall be performed as specified in Annex A, which defines CBC encryption in the general sense. A security suite implementing CBC encryption will specify any parameters left unspecified in Annex A according to the requirements of that particular security suite.

#### **1.4.1.7 CBC-MAC authentication**

The AES cipher block chaining message authentication code (CBC-MAC) symmetric authentication algorithm used in this standard consists of the generation of an integrity code using a block cipher in CBC mode computed on a message that includes the length of the authenticated data at the beginning of the data itself. The verification operation consists of the computation of this integrity code and comparison to the received integrity code.

All of the above operations shall be performed as specified in Annex A, which defines CBC-MAC authentication in the general sense. A security suite implementing CBC-MAC authentication will specify any parameters left unspecified in Annex A according to the requirements of that particular security suite.

#### **1.4.1.8 CTR + CBC-MAC (CCM) combined encryption and authentication**

The AES counter mode encryption plus cipher block chaining message authentication code (CCM) combined symmetric encryption and authentication mechanism used in this standard consists of the generation of an integrity code followed by the encryption of plaintext data and the integrity code. The output consists of the encrypted data and the encrypted integrity code.

The symmetric authentication operation used in this security suite consists of the generation of an integrity code using a block cipher in CBC mode computed on a nonce followed by padded authentication data followed by padded plaintext data, if present. The verification operation consists of the computation of this integrity code and comparison to the received integrity code.

The symmetric encryption operation used in this security suite consists of the generation of a key stream using a block cipher in counter mode with a given key and nonce and performing an exclusive-OR of the key stream with the integrity code and plaintext, if present. The decryption operation consists of the generation of the key stream and the exclusive-OR of the key stream with the ciphertext to obtain the plaintext and integrity code.

All of the above operations shall be performed as specified in Annex A, which defines CCM mode in the general sense. A security suite implementing CCM combined encryption and authentication will specify any parameters left unspecified in Annex A according to the requirements of that particular security suite.

#### **1.4.1.9 AES encryption**

The advanced encryption standard (AES) Rijndael encryption algorithm used in this standard shall be performed as specified in the FIPS 197 standard. This encryption algorithm is parameterized by the use of a 128-bit block size and a key length of 128-bits for each security suite described in this standard.

#### **1.4.1.10 PIB security material**

This subclause describes the formats of the security information stored in the MAC PIB. This information is dependent on the individual security suite selected and is referenced in the following subclauses.

The symmetric key is the 128-bit AES key for this ACL entry that shall be used to perform exactly one of (ECB encryption) OR (CTR encryption) OR (CBC encryption) OR (CCM encryption and authentication) OR (CBC-MAC authentication). An AES key shall not be used with different security suites.

The initialization vector is a pseudo-random initializing variable to be used with the CBC encryption security suite, providing immediate randomness and security in the first chaining stage. The generation of the IV is implementation specific and outside the scope of this document; however, the IV shall be unique for each message encrypted or decrypted with a given symmetric key.

The frame counter is the running counter that shall be included in the payload field in the MAC payload of the MAC frame. This counter is incremented each time a secure frame is transmitted, as specified in the outgoing frame operations' sections in security suites that use frame counters. This counter will not roll over (see 1.4.1.3). This value helps to ensure that the CCM nonce is unique and allows the recipient to use the counter to ensure freshness.

The key sequence counter is a counter that is fixed by the higher layer that shall be included in the payload field in the MAC payload of the MAC frame. The key sequence counter can be used, for example, if the frame counter is exhausted. This value helps to ensure that the CCM nonce is unique and allows the recipient to use the counter to ensure freshness. If freshness is used, the higher layer should not decrease this counter as the freshness operation on the recipient side would fail.

The optional external frame counter and optional external key sequence counter are fields that may be stored in the ACL entry that represent the values of the last received frame counter and key sequence counter respectively in secure frames corresponding to this ACL entry. If the optional external frame counter and optional external key sequence counter fields are included in the ACL entry, the MAC will use them to verify the sequential freshness of received secure frames as described in the 'incoming frame operations' sections in security suites that use frame counters.

## 1.4.2 AES ECB security suite

The AES ECB security suite is used when a device is operating in secured mode. The cryptographic operations in this security suite consist of performing AES ECB encryption (or decryption) on the payload field within the MAC payload using a 16-octet key.

The AES ECB security suite provides the following security services:

- Access control
- Data encryption
- Sequential freshness (optional)

### 1.4.2.1 Data formats

The following subclauses define the data formats used in this security suite.

#### 1.4.2.1.1 MAC PIB formats

For the AES ECB security suite, the security material stored in *macDefaultSecurityMaterial* or the *ACLSecurityMaterial* field in the ACL consists of a symmetric key, a frame counter and a key sequence counter as well as an optional frame counter and sequence counter that may be used for incoming frames. If these optional fields are included in the security material, the optional operations specified in 1.4.2.3.2 shall be performed. When these optional operations are performed, the security suite provides the sequential freshness security service on received frames. Figure 1 below shows the order and length of the AES ECB security material components.

<b>Octets: 16</b>	<b>4</b>	<b>1</b>	<b>(4)</b>	<b>(1)</b>
Symmetric key	Frame counter	Key sequence counter	Optional external frame counter	Optional external key sequence counter

**Figure 1 – AES ECB security material**

#### 1.4.2.1.2 Protected payload field formats

In the AES ECB security suite, the payload field in the MAC payload of a protected frame consists of the frame counter, the key sequence counter, and the encrypted payload. Figure 2 below specifies the order and length of the subfields of the payload field of an AES ECB secured frame. The length of the encrypted payload field is equal to the length of the payload field before it was encrypted.

<b>Octets: 4</b>	<b>1</b>	<b>variable</b>
Frame counter	Key sequence counter	Encrypted payload

**Figure 2 – AES ECB payload field**

#### 1.4.2.1.3 ECB input blocks

In the AES ECB security suite, the forward cipher function is invoked on each 128-bit block of plaintext data, using a 128-bit key, to produce the 128-bit ciphertext block, and vice versa. If a partial block of plaintext data is to be encrypted, the partial block should be left-justified and padded with zeroes to complete the 128-bit plaintext block. Ciphertext to be decrypted is therefore guaranteed to be block-aligned. The input block to the ECB cipher is shown in the figure below.

<b>Octets: 16</b>
Cipher input data

**Figure 3 – AES ECB input block**

#### 1.4.2.2 Security parameters

The AES ECB encryption operation in this security suite, as defined in 1.4.1.4, shall be parameterized by the following: the underlying block cipher shall be the AES encryption algorithm as specified in 1.4.1.9, the ECB input blocks shall be formed as specified in 1.4.2.1.3 where any partial input plaintext block shall be left-justified and padded with zeroes to complete the 128-bit input block. See Annex A for further explanation.

#### 1.4.2.3 Security operations

The following subclauses specify the operations performed on outgoing and incoming frames.

#### 1.4.2.3.1 Outgoing frame operations

When the AES ECB security suite is invoked to protect an outgoing frame, the MAC sublayer shall perform the following operations in order:

1. Obtain its own 64-bit extended address, *aExtendedAddress*, along with the symmetric key, the frame counter and the sequence counter from the MAC PIB, and construct the ECB input blocks as specified in 1.4.2.1.3.
2. Encrypt the payload field in the MAC payload of the frame using ECB encryption, as specified in 1.4.1.4, with the parameters specified in 1.4.2.2 and using the input blocks from step 1.
3. Combine the frame counter, sequence counter, and output from step 2, as specified in 1.4.2.1.2, to obtain the new payload field.
4. Increment the frame counter as specified in 1.4.1.3 and, if the incrementing succeeds, insert the new counter value into the MAC PIB. If the incrementing operation fails due to the counter value rolling over, the device shall abort the operation and notify the higher layer.

#### 1.4.2.3.2 Incoming frame operations

When the AES ECB security suite is invoked to protect an incoming frame, the MAC sublayer shall perform the following operations in order:

1. If the optional external frame and external key sequence counters are included in the corresponding *macDefaultSecurityMaterial* or *ACLSecurityMaterial* field, ensure sequential freshness by verifying that the received key sequence counter is greater than or equal to the external key sequence counter from that device. If the key sequence counter is greater than or equal to the external key sequence counter, verify that the received frame counter is greater than or equal to the external frame counter from that device. If either of these checks fail, the device shall reject the frame and issue an indication to the higher layer.
2. Extract the frame counter and sequence counter from the payload field in the MAC payload and construct the ECB input blocks as specified in 1.4.2.1.3.
3. Decrypt the encrypted payload field using ECB decryption, as specified in 1.4.1.4, with the parameters specified in 1.4.2.2 and using the ECB input blocks from step 2 above.
4. Replace the existing payload field in the MAC payload with the decrypted data from step 3. If optional operation 1 was performed, and the checks succeeded, the last known sequence counter and last known frame counter shall be set to the received values.

### 1.4.3 AES CTR security suite

The AES CTR security suite is used when a device is operating in secured mode. The cryptographic operations in this security suite consist of performing AES CTR encryption (or decryption) on the payload field within the MAC payload using shared data, the frame counter and the key sequence counter.

The AES CTR security suite provides the following security services:

- Access control
- Data encryption
- Sequential freshness (optional)

#### 1.4.3.1 Data formats

The following subclauses define the data formats used in this security suite.

##### 1.4.3.1.1 MAC PIB formats

For the AES CTR security suite, the security material stored in *macDefaultSecurityMaterial* or the *ACLSecurityMaterial* field in the ACL consists of a symmetric key, a frame counter and a key sequence counter as well as an optional frame counter and sequence counter that may be used for incoming frames. If these optional fields are included in the security material, the optional operations specified in 1.4.3.3.2 shall be performed. When these optional operations are performed, the security suite provides the sequential freshness security service on received frames. Figure 4 below shows the order and length of the AES CTR security material components.

<b>Octets: 16</b>	<b>4</b>	<b>1</b>	<b>(4)</b>	<b>(1)</b>
Symmetric key	Frame counter	Key sequence counter	Optional external frame counter	Optional external key sequence counter

**Figure 4 – AES CTR security material**

### 1.4.3.1.2 Protected payload field formats

In the AES CTR security suite, the payload field in the MAC payload of a protected frame consists of the frame counter, the key sequence counter, and the encrypted payload. Figure 5 below specifies the order and length of the subfields of the payload field of an AES CTR secured frame. The length of the encrypted payload field is equal to the length of the payload field before it was encrypted.

<b>Octets: 4</b>	<b>1</b>	<b>variable</b>
Frame counter	Key sequence counter	Encrypted payload

**Figure 5 – AES CTR payload field**

### 1.4.3.1.3 CTR input blocks

In the AES CTR security suite, the input blocks to the CTR encryption function for generating the key stream consist of a flag octet, the address of the device sending the frame, the frame and key sequence counter values and the block counter value. Figure 6 below specifies the order and length of the subfields of the CTR input blocks, the use of which is described in 1.4.3.3.1 and 1.4.3.3.2. These input blocks correspond to the counters  $T_1, T_2, \dots, T_n$  as specified in Annex A.

<b>Octets: 1</b>	<b>8</b>	<b>4</b>	<b>1</b>	<b>2</b>
Flags	Source address	Frame counter	Key sequence counter	Block counter

**Figure 6 – AES CTR input block**

The flags octet used in AES CTR mode, which is used as padding for the input block, is formatted as specified in Figure 7 below. The bit values in this flag octet are chosen to distinguish the AES CTR flags from the AES CCM flags.

<b>Bits: 7</b>	<b>6-2</b>	<b>1</b>	<b>0</b>
1	0	1	0

**Figure 7 – AES CTR flags field**

### 1.4.3.2 Security parameters

The AES CTR encryption operation in this security suite, as defined in 1.4.1.5, shall be parameterized by the following: the underlying block cipher shall be the AES encryption algorithm as specified in 1.4.1.9, the counter input blocks shall be formed as specified in 1.4.3.1.3 where each input block is the same except that the block counter is set to 0 for

the first block and is incremented as specified in 1.4.1.3 for each successive input block. In other words, the  $i^{\text{th}}$  input block  $T_i$  shall have the block counter value set to  $i-1$ .

### 1.4.3.3 Security operations

The following subclauses specify the operations performed on outgoing and incoming frames.

#### 1.4.3.3.1 Outgoing frame operations

When the AES CTR security suite is invoked to protect an outgoing frame, the MAC sublayer shall perform the following operations in order:

1. Obtain its own 64-bit extended address, *aExtendedAddress*, along with the symmetric key, the frame counter and the sequence counter from the MAC PIB, and construct the counter input blocks as specified in 1.4.3.1.3.
2. Encrypt the payload field in the MAC payload of the frame using CTR encryption, as specified in 1.4.1.5, with the parameters specified in 1.4.3.2 and using the input blocks from step 1.
3. Combine the frame counter, sequence counter, and output from step 2, as specified in 1.4.3.1.2, to obtain the new payload field.
4. Increment the frame counter as specified in 1.4.1.3 and, if the incrementing succeeds, insert the new counter value into the MAC PIB. If the incrementing operation fails due to the counter value rolling over, the device shall abort the operation and notify the higher layer.

#### 1.4.3.3.2 Incoming frame operations

When the AES CTR security suite is invoked to protect an incoming frame, the MAC sublayer shall perform the following operations in order:

1. If the optional external frame and external key sequence counters are included in the corresponding *macDefaultSecurityMaterial* or *ACLSecurityMaterial* field, ensure sequential freshness by verifying that the received key sequence counter is greater than or equal to the external key sequence counter from that device. If the key sequence counter is greater than or equal to the external key sequence counter, verify that the received frame counter is greater than or equal to the external frame counter from that device. If either of these checks fail, the device shall reject the frame and issue an indication to the higher layer.
2. Obtain the 64-bit extended address of the source either from the frame or from the ACL, extract the frame counter and sequence counter from the payload field in the MAC payload and construct the counter input blocks as specified in 1.4.3.1.3. If the input block can not be constructed due to any data being unavailable, the device shall issue an indication to the higher layers.

3. Decrypt the encrypted payload field using CTR decryption, as specified in 1.4.1.5, with the parameters specified in 1.4.3.2 and using the counter input blocks from step 2 above.
4. Replace the existing payload field in the MAC payload with the decrypted data from step 3. If optional operation 1 was performed, and the checks succeeded, the last known sequence counter and last known frame counter shall be set to the received values.

#### 1.4.4 AES CBC security suite

The AES CBC security suite is used when a device is operating in secured mode. The cryptographic operations in this security suite consist of performing AES CBC encryption (or decryption) on the payload field within the MAC payload using shared data, the initialization vector (IV), the frame counter and the key sequence counter.

The AES CBC security suite provides the following security services:

- Access control
- Data encryption
- Sequential freshness (optional)

##### 1.4.4.1 Data formats

The following subclauses define the data formats used in this security suite.

###### 1.4.4.1.1 MAC PIB formats

For the AES CBC security suite, the security material stored in *macDefaultSecurityMaterial* or the *ACLSecurityMaterial* field in the ACL consists of a symmetric key, an IV, a frame counter and a key sequence counter as well as an optional frame counter and sequence counter that may be used for incoming frames. If these optional fields are included in the security material, the optional operations specified in 1.4.4.3.2 shall be performed. When these optional operations are performed, the security suite provides the sequential freshness security service on received frames. Figure 8 below shows the order and length of the AES CBC security material components.

Octets: 16	16	4	1	(4)	(1)
Symmetric key	Initialization vector	Frame counter	Key sequence counter	Optional external frame counter	Optional external key sequence counter

**Figure 8 – AES CBC security material**

#### 1.4.4.1.2 Protected payload field formats

In the AES CBC security suite, the payload field in the MAC payload of a protected frame consists of the IV, the frame counter, the key sequence counter, and the encrypted payload. Figure 9 below specifies the order and length of the subfields of the payload field of an AES CBC secured frame. The length of the encrypted payload field is equal to the length of the payload field before it was encrypted.

<b>Octets: 16</b>	<b>4</b>	<b>1</b>	<b>variable</b>
Initialization vector	Frame counter	Key sequence counter	Encrypted payload

**Figure 9 – AES CBC payload field**

#### 1.4.4.1.3 CBC input blocks

In the AES CBC security suite, a feedback mechanism is added to the block cipher. Encryption consists of exclusive-ORing the 128-bit block of current plaintext with the previous ciphertext block and then encrypting the 128-bit exclusive-OR output, using a 128-bit key. Decryption consists of operating on the 128-bit block of current ciphertext, using the 128-bit key, then exclusive-ORing the current cipher output with the previous ciphertext block to produce a 128-bit block of plaintext. A 128-bit initialization vector (IV) is used to fill the feedback register for the first block to be processed. Refer to Annex A for further details. The inputs to the first chaining block are shown in Figure 10 below, with the input data format for consecutive stages shown in Figure 11.

<b>Octets: 16</b>	<b>16</b>
Initialization vector	Cipher input data

**Figure 10 – AES CBC first stage inputs**

<b>Octets: 16</b>
Cipher input data

**Figure 11 – AES CBC consecutive stage input data**

For the last block, which may be a partial block of  $u$  bits, the most significant  $u$  bits of the last input plaintext block are left-justified; the remaining  $b-u$  bits are zero-padded and concatenated with the  $u$  bits to form a complete block for input to the exclusive-OR function feeding the forward cipher. Refer to Annex A for further details.

#### 1.4.4.2 Security parameters

The AES CBC encryption operation in this security suite, as defined in 1.4.1.6, shall be parameterized by the following: the underlying block cipher shall be the AES encryption algorithm as specified in 1.4.1.9, the CBC input blocks shall be formed as specified in 1.4.4.1.3 where the first stage includes exclusive-ORing of the IV for immediate randomization, and consecutive stages include chaining feedback from the previous stage. A graphical representation is shown in Annex A Figure A.3.

#### 1.4.4.3 Security operations

The following subclauses specify the operations performed on outgoing and incoming frames.

##### 1.4.4.3.1 Outgoing frame operations

When the AES CBC security suite is invoked to protect an outgoing frame, the MAC sublayer shall perform the following operations in order:

1. Obtain its own 64-bit extended address, *aExtendedAddress*, along with the symmetric key, the IV, the frame counter and the sequence counter from the MAC PIB, and construct the CBC input blocks as specified in 1.4.4.1.3.
2. Encrypt the payload field in the MAC payload of the frame using CBC encryption, as specified in 1.4.1.6, with the parameters specified in 1.4.4.2 and using the input blocks from step 1.
3. Combine the IV, frame counter, sequence counter, and output from step 2, as specified in 1.4.4.1.2, to obtain the new payload field.
4. Increment the frame counter as specified in 1.4.1.3 and, if the incrementing succeeds, insert the new counter value into the MAC PIB. If the incrementing operation fails due to the counter value rolling over, the device shall abort the operation and notify the higher layer.

##### 1.4.4.3.2 Incoming frame operations

When the AES CBC security suite is invoked to protect an incoming frame, the MAC sublayer shall perform the following operations in order:

1. If the optional external frame and external key sequence counters are included in the corresponding *macDefaultSecurityMaterial* or *ACLSecurityMaterial* field, ensure sequential freshness by verifying that the received key sequence counter is greater than or equal to the external key sequence counter from that device. If the key sequence counter is greater than or equal to the external key sequence counter, verify that the received frame counter is greater than or equal to the

- external frame counter from that device. If either of these checks fail, the device shall reject the frame and issue an indication to the higher layer.
2. Extract the IV, frame counter and sequence counter from the payload field in the MAC payload and construct the CBC input blocks as specified in 1.4.4.1.3.
  3. Decrypt the encrypted payload field using CBC decryption, as specified in 1.4.1.6, with the parameters specified in 1.4.4.2 and using the CBC input blocks from step 2 above.
  4. Replace the existing payload field in the MAC payload with the decrypted data from step 3. If optional operation 1 was performed, and the checks succeeded, the last known sequence counter and last known frame counter shall be set to the received values.

### **1.4.5 AES CCM security suite**

The AES CCM security suite is used when a device is operating in secured mode. The cryptographic operations in this security suite consist of performing AES CCM authentication (or verification) on the MAC header concatenated with the MAC payload and encryption (or decryption) on the payload field within the MAC payload using shared data, the frame counter and the key sequence counter. The AES CCM security suite shall be implemented using 128-bit integrity codes.

The AES CCM security suite provides the following security services:

- Access control
- Data encryption
- Frame integrity
- Sequential freshness (optional)

#### **1.4.5.1 Data formats**

The following subclauses define the data formats used in this security suite.

##### **1.4.5.1.1 MAC PIB formats**

For the AES CCM security suite, the security material stored in *macDefaultSecurityMaterial* or the *ACLSecurityMaterial* field in the ACL consists of a symmetric key, a frame counter and a key sequence counter as well as an optional frame counter and sequence counter that may be used for incoming frames. If these optional fields are included in the security material, the optional operations specified in 1.4.5.3.2 shall be performed. When these optional operations are performed, the security suite provides the sequential freshness security service on received frames. Figure 12 below shows the order and length of the AES CCM security material components.

<b>Octets: 16</b>	<b>4</b>	<b>1</b>	<b>(4)</b>	<b>(1)</b>
Symmetric key	Frame counter	Key sequence counter	Optional external frame counter	Optional external key sequence counter

**Figure 12 – AES CCM security material**

#### 1.4.5.1.2 Protected payload field formats

In the AES CCM security suite, the payload field in the MAC payload of a protected frame consists of the frame counter, the key sequence counter, the encrypted payload and the encrypted integrity code. Figure 13 below specifies the order and length of the subfields of the payload field of an AES CCM secured frame. The length of the encrypted payload field is equal to the length of the payload field before it was encrypted. The length of the encrypted integrity code subfield is equal to the length of the integrity code before it was encrypted, i.e. 16 octets.

<b>Octets: 4</b>	<b>1</b>	<b>variable</b>	<b>16</b>
Frame counter	Key sequence counter	Encrypted payload	Encrypted integrity code

**Figure 13 – AES CCM payload field**

#### 1.4.5.1.3 CCM nonce

In the AES CCM security suite, the nonce input used for the CCM authentication and encryption function consists of data explicitly included in the frame and data that both devices can independently obtain. Figure 14 below specifies the order and length of the subfields of the CCM nonce.

<b>Octets: 8</b>	<b>4</b>	<b>1</b>
Source address	Frame counter	Key sequence counter

**Figure 14 – AES CCM nonce**

#### 1.4.5.2 Security parameters

In this security suite, the CCM operations as defined in 1.4.1.8 shall be parameterized by the following: the underlying block cipher shall be the AES encryption algorithm as specified in 1.4.1.9, the length in octets of the length field L shall be 2 octets, the length of the authentication field M shall be 16 octets and the nonce shall be formatted as specified in 1.4.5.1.3.

### 1.4.5.3 Security operations

The following subclauses specify the operations performed on outgoing and incoming frames.

#### 1.4.5.3.1 Outgoing frame operations

When the AES CCM security suite is invoked to protect an outgoing frame, the MAC sublayer shall perform the following operations in order:

1. Obtain its own 64-bit extended address, *aExtendedAddress*, along with the symmetric key, the frame counter and the sequence counter from the MAC PIB, and construct the nonce as specified in 1.4.5.1.3.
2. Encrypt and authenticate the MAC header and MAC payload in the frame using CCM authentication and encryption, as specified in 1.4.1.8, with the parameters specified in 1.4.5.2. Use the MAC header and non-payload fields in the MAC payload as the authentication data, *a*, the payload field in the MAC payload as the message, *m*, and the nonce computed in step 1.
3. Combine the frame counter, sequence counter, and output from step 2 (including the encrypted payload and encrypted integrity code), as specified in 1.4.5.1.2, to obtain the new payload field.
4. Increment the frame counter as specified in 1.4.1.3 and, if the incrementing succeeds, insert the new counter value into the MAC PIB. If the incrementing operation fails due to the counter value rolling over, the device shall abort the operation and notify the higher layer.

#### 1.4.5.3.2 Incoming frame operations

When the AES CCM security suite is invoked to protect an incoming frame, the MAC sublayer shall perform the following operations in order:

1. If the optional external frame and external key sequence counters are included in the corresponding *macDefaultSecurityMaterial* or *ACLSecurityMaterial* field, ensure sequential freshness by verifying that the received key sequence counter is greater than or equal to the external key sequence counter from that device. If the key sequence counter is greater than or equal to the external key sequence counter, verify that the received frame counter is greater than or equal to the external frame counter from that device. If either of these checks fail, the device shall reject the frame and issue an indication to the higher layer.
2. Obtain the 64-bit extended address of the source either from the frame or from the ACL, remove the frame counter and sequence counter from the payload field in the MAC payload and construct the nonce as specified in 1.4.5.1.3. If the nonce can not be constructed due to any data being unavailable, the device shall issue an indication to the higher layers.

3. Decrypt the encrypted payload field and verify the integrity code using CCM decryption and authentication, as specified in 1.4.1.8, with the parameters specified in 1.4.5.2. Use the MAC header and non-payload fields in the MAC payload as the authentication data,  $a$ , the encrypted payload field as the message,  $m$ , and the nonce computed in step 2. If the integrity code fails, the device shall discard the frame and issue an indication to the higher layers.
4. Replace the existing payload field in the MAC payload with the decrypted data from step 3. If optional operation 1 was performed, and the checks succeeded, the last known sequence counter and last known frame counter shall be set to the received values.

### 1.4.6 AES CBC-MAC security suite

The AES CBC-MAC security suite is used when a device is operating in secured mode. The cryptographic operations in this security suite consist of performing AES CBC-MAC authentication on the MAC header and MAC payload. The AES CBC-MAC security suite shall be implemented using 128-bit integrity codes.

The AES CBC-MAC security suite provides the following security services:

- Access control
- Frame integrity

#### 1.4.6.1 Data formats

The following subclauses define the data formats used in this security suite.

##### 1.4.6.1.1 MAC PIB formats

For the AES CBC-MAC security suite, the security material stored in *macDefaultSecurityMaterial* or the *ACLSecurityMaterial* field in the ACL consists of a symmetric key. No state information is required between frames. Figure 15 below shows the order and length of the AES CBC-MAC security material components.

<b>Octets: 16</b>
Symmetric key

**Figure 15 – AES CBC-MAC security material**

### 1.4.6.1.2 Protected payload field formats

In the AES CBC-MAC security suite, the payload field in the MAC payload of a protected frame consists of the existing payload followed by the integrity code. Figure 16 below specifies the order and length of the subfields of the payload field of an AES CBC-MAC secured frame. The length of the integrity code subfield is 16 octets.

<b>Octets: variable</b>	<b>16</b>
Payload	Integrity code

**Figure 16 – AES CBC-MAC payload field**

### 1.4.6.1.3 CBC-MAC input blocks

In the AES CBC-MAC security suite, the input to the CBC-MAC authentication function for generating the integrity code consists of the length of the data to authenticate (not including the length field itself) followed by the MAC header followed by the MAC payload. The input is broken up into 16-octet blocks starting from the left and proceeding to the right until the last block, which may be smaller than 16 octets depending on the length of the total input. If the number of data bits is not a multiple of the block size, then the final input block will be a partial block of data, left justified, with zeroes appended to form a full block. Figure 17 below specifies the order and length of the subfields of the CBC-MAC input.

<b>Octets: 1</b>	<b>variable = n</b>	<b>variable = m</b>
Length = n + m	MAC header	MAC payload

**Figure 17 – AES CBC-MAC input**

### 1.4.6.2 Security parameters

In this security suite, the CBC-MAC operations as defined in 1.4.1.7 shall be parameterized by the following: the underlying block cipher shall be the AES encryption algorithm as specified in 1.4.1.9, the input to the CBC-MAC function shall be formatted as specified in 1.4.6.1.3, and the length of the integrity code M shall be 128 bits.

### 1.4.6.3 Security operations

The following subclauses specify the operations performed on outgoing and incoming frames.

#### **1.4.6.3.1 Outgoing frame operations**

When the AES CBC-MAC security suite is invoked to protect an outgoing frame, the MAC sublayer shall perform the following operations in order:

1. Determine the length in octets of the MAC header concatenated with the MAC payload (before the security operations are performed) and encode that length as a 1-octet integer, as specified in 1.4.1.3.
2. Compute the integrity code on the MAC header and MAC payload in the frame using CBC-MAC authentication, as specified in 1.4.1.7, with the parameters specified in 1.4.6.2.
3. Combine the existing payload field in the MAC payload and output from step 2, as specified in 1.4.6.1.2, to obtain the new payload field.

#### **1.4.6.3.2 Incoming frame operations**

When the AES CBC-MAC security suite is invoked to protect an incoming frame, the MAC sublayer shall perform the following operations in order:

1. Determine the length in octets of the MAC header concatenated with the MAC payload (before the security operations are applied) and encode that length as a 1-octet integer, as specified in 1.4.1.3.
2. Parse the payload field of the MAC payload into the payload and integrity code subfields, as specified in 1.4.6.1.2, and verify the integrity code using CBC-MAC authentication, as specified in 1.4.1.7, with the parameters specified in 1.4.6.2. Use the length determined in step 1, the MAC header from the received frame and the MAC payload (without the integrity code) as the input data. If the integrity code fails, the device shall discard the frame and notify the higher layers.
3. Remove the integrity code from the payload field in the MAC payload.

# Annex A: Security implementation

## A.1 AES ECB Mode Operation

The Electronic Codebook mode (ECB) is the most straightforward way to employ the AES block cipher. Encryption consists of operating on a 128-bit block of plaintext, using a 128-bit key, to produce a 128-bit block of ciphertext. Decryption consists of operating on a 128-bit block of ciphertext, using the 128-bit key, to produce a 128-bit block of plaintext.

ECB Encryption:  $C_j = E_k(P_j)$  for  $j = 1, 2 \dots n$ ;

ECB Decryption:  $P_j = D_k(C_j)$  for  $j = 1, 2 \dots n$ ;

In ECB encryption, the forward cipher function is invoked on each 128-bit block of plaintext data, using a 128-bit key, to produce the 128-bit ciphertext block. If a partial block of plaintext data is to be encrypted, the partial block should be left-justified and padded with zeroes to a complete the 128-bit plaintext block.

In ECB decryption, the ~~forward~~-reverse cipher function is invoked on each 128-bit block of ciphertext data, using a 128-bit key, to produce the 128-bit plaintext block.

In both ECB encryption and ECB decryption, the ~~forward~~-cipher functions can be performed in parallel, and the efficiency of ECB is limited only by the speed of the block cipher. AES ECB is the most straightforward mode of AES to implement and may be useful for a range of low-cost and/or compact applications using this standard.

The structure of AES ECB mode is graphically represented in the figure below.

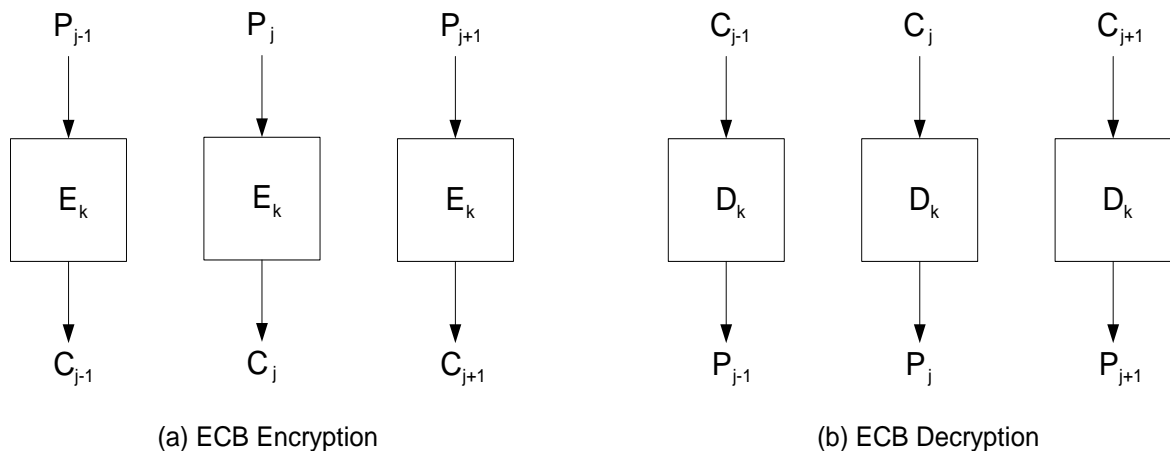


Figure A.1 – AES ECB Mode

## A.2 AES CTR Mode Operation

The Counter (CTR) mode provides data encryption that applies the forward cipher to a set of input blocks, called counters, to produce a sequence of output blocks that are exclusive-ORed with the plaintext to produce the ciphertext, and vice versa for decryption. The sequence of counters must have the property that each block in the sequence is unique. Across all messages that are encrypted under the given 128-bit key, all of the counters must be different from each other. Let the counters for a given message be denoted  $T_1, T_2, \dots, T_n$ . Given a sequence of counters,  $T_1, T_2, \dots, T_n$ , the CTR mode is defined as follows:

$$\begin{aligned} \text{CTR Encryption:} \quad C_j &= P_j \text{ XOR } E_k(T_j) && \text{for } j = 1, 2 \dots n-1; \\ C_n^* &= P_n^* \text{ XOR } \text{MSB}_u(E_k(T_n)) \end{aligned}$$

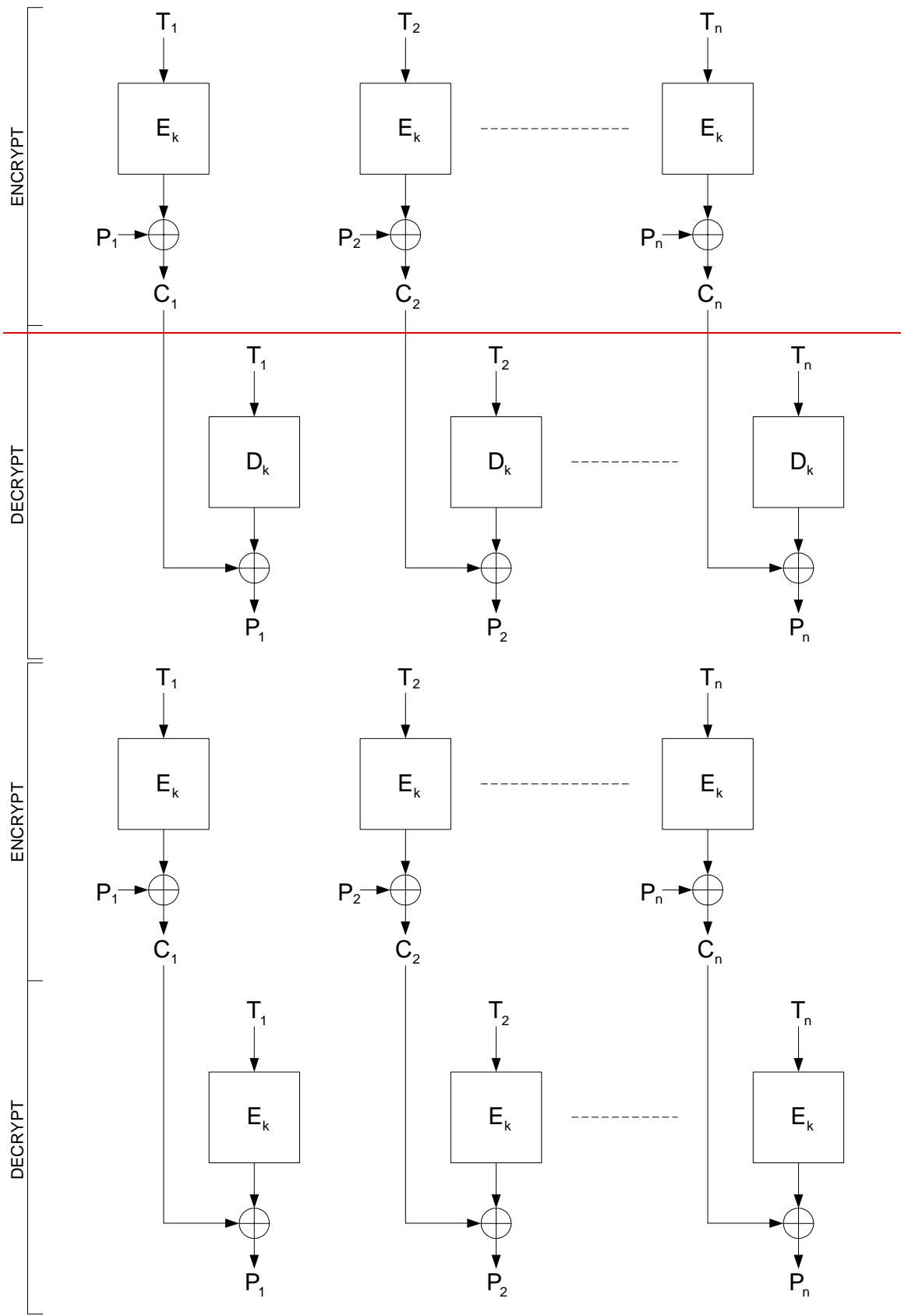
$$\begin{aligned} \text{CTR Decryption:} \quad P_j &= C_j \text{ XOR } E_k(T_j) && \text{for } j = 1, 2 \dots n-1; \\ P_n^* &= C_n^* \text{ XOR } \text{MSB}_u(E_k(T_n)) \end{aligned}$$

In CTR encryption, the forward cipher function is invoked on each counter block, and the resulting output blocks are exclusive-ORed with the corresponding plaintext blocks to produce the ciphertext blocks. For the last block, which may be a partial block of  $u$  bits, the most significant  $u$  bits of the last output block are used for the exclusive-OR operation; the remaining  $b-u$  bits of the last output block are discarded, where  $b$  is the length in bits of the block cipher.

In CTR decryption, the forward cipher function is invoked on each counter block, and the resulting output blocks are exclusive-ORed with the corresponding ciphertext blocks to recover the plaintext blocks. For the last block, which may be a partial block of  $u$  bits of the last output block are used for the exclusive-OR operation; the remaining  $b-u$  bits of the last output block are discarded.

In both CTR encryption and CTR decryption, the forward cipher functions can be performed in parallel. Plaintext recovery of any particular block is independent of any other plaintext block and can be achieved if the corresponding counter block can be determined. Furthermore, pre-processing of forward cipher functions is possible prior to the availability of the plaintext or ciphertext data.

The structure of AES CTR mode is graphically represented in the figure below.



## Figure A.2 – AES CTR Mode

### A.3 AES CBC Mode Operation

In Cipher Block Chaining mode (CBC), a feedback mechanism is added to the block cipher. Encryption consists of exclusive-ORing the 128-bit block of current plaintext with the previous ciphertext block and then encrypting the 128-bit exclusive-OR output, using a 128-bit key. Decryption consists of operating on the 128-bit block of current ciphertext, using the 128-bit key, then exclusive-ORing the current cipher output with the previous ciphertext block to produce a 128-bit block of plaintext. A 128-bit initialization vector (IV) is used to fill the feedback register for the first block to be processed (see Figure A.3 below).

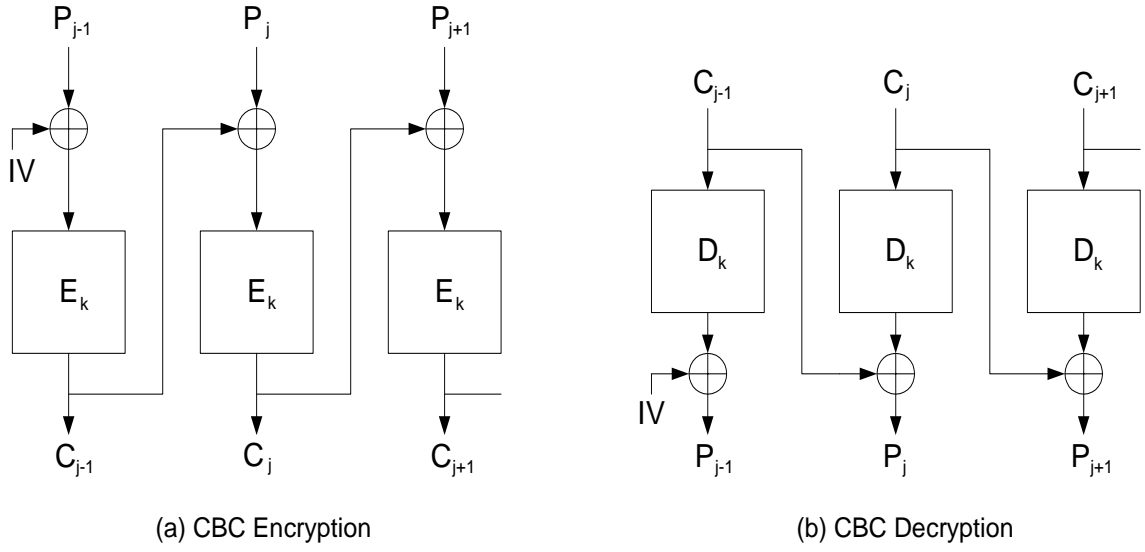
$$\begin{aligned} \text{CBC Encryption:} \quad C_1 &= E_k(P_1 \text{ XOR IV}) \\ C_j &= E_k(P_j \text{ XOR } C_{j-1}) \quad \text{for } j = 2, 3, \dots, n \end{aligned}$$

$$\begin{aligned} \text{CBC Decryption:} \quad P_1 &= IV \text{ XOR } D_k(C_1) \\ P_j &= C_{j-1} \text{ XOR } D_k(C_j) \quad \text{for } j = 2, 3, \dots, n \end{aligned}$$

In CBC encryption, the plaintext is exclusive-ORed with the previous ciphertext block, then the exclusive-ORed output is encrypted. For the last block, which may be a partial block of  $u$  bits, the most significant  $u$  bits of the last input plaintext block are left-justified; the remaining  $b-u$  bits are zero-padded and concatenated with the  $u$  bits to form a complete block for input to the exclusive-OR function feeding the forward cipher.

Plaintext patterns are concealed and input to the block cipher is randomized by the exclusive-ORing function, which increases the level of security. Due to the structure of the serial chaining path, CBC encryption is not parallelizable; however, CBC decryption is parallelizable and has a random-access property.

The structure of AES CBC mode is graphically represented in the figure below.



**Figure A.3 – AES CBC Mode**

## A.4 AES CCM Mode Operation

CCM mode consists of CTR mode encryption combined with CBC-MAC authentication to produce an authenticate-and-encrypt block cipher mode. CCM is defined here and intended for use with 128-bit block size AES.

For the generic CCM mode there are two parameter choices to be made. The first choice is  $M$ , the size of the authentication field. The choice of the value for  $M$  involves a trade-off between message expansion and the probability that an attacker can undetectably modify a message. Valid values are 4, 6, 8, 10, 12, 14, and 16 octets. The second choice is  $L$ , the size of the length field. This value requires a trade-off between the maximum message size and the size of the Nonce. Different applications require different trade-offs, so  $L$  is a parameter. Valid values are 2 to 8 octets, where the value  $L=1$  is reserved. The CCM mode parameters are shown in the table below.

Name	Description	Field size	Encoding of field
$M$	Number of octets in authentication field	3 bits	$(M-2)/2$
$L$	Number of octets in length field	3 bits	$L-1$

**Table A.1 – Parameters of CCM mode**

### A.4.1 Inputs

To send a message the sender must provide the following information:

- A 128-bit encryption key  $K$  for the AES block cipher.
- A nonce  $N$  of  $15-L$  octets. For any given encryption key  $K$ , the nonce value shall be unique. Using the same nonce for two different messages encrypted with the same key destroys the security properties of this mode.
- The message  $m$ , consisting of a string  $l(m)$  octets where  $0 \leq l(m) < 2^{8L}$ . This length restriction ensures that  $l(m)$  can be encoded in a field of  $L$  octets.
- Additional authenticated data  $a$ , consisting of a string of  $l(a)$  octets where  $0 \leq l(a) < 2^{64}$ . This additional data is authenticated but not encrypted, and is not included in the output of CCM. It can be used to authenticate plaintext headers, or contextual information that affects the interpretation of message. It is application specific. Users who do not wish to authenticate additional data can provide a string of length zero.

This information is summarized in the table below.

Name	Description	Field size	Encoding of field
$K$	AES block cipher key	16 octets	String of octets
$N$	Nonce	15- $L$ octets	Not specified
$m$	Message to be encrypted and sent	$l(m)$ octets	String of octets
$a$	Additional authenticated data	$l(a)$ octets	String of octets

**Table A.2 – Inputs for CCM**

#### A.4.2 Authentication

The first step is to compute the authentication field  $T$ . This is accomplished using CBC-MAC. We first define a sequence of blocks  $B_0, B_1, \dots, B_n$  and then apply CBC-MAC to these blocks. The first block  $B_0$  is formatted as indicated in the table below.

<b>Octet #:</b>	0	1 ... 15- $L$	16- $L$ ... 15
<b>Contents:</b>	Flags	Nonce $N$	$l(m)$

**Table A.3 – First authentication block  $B_0$**

The value  $l(m)$  is encoded in most-significant-octet first order. The **Flags** field is formatted as indicated in the table below.

<b>Bit #:</b>	7	6	5	4	3	2	1	0
<b>Contents:</b>	<i>Reserved</i>	Adata	M			L		

**Table A.4 – Authentication flags octet**

The Reserved bit should always be set to zero. The Adata bit is set to zero if  $l(a)=0$ , and set to one if  $l(a)>0$ . The  $M$  field is assigned the value of  $4*(\text{bit } 5) + 2*(\text{bit } 4) + (\text{bit } 3)$  and encodes the value of  $M$  as  $(M-2)/2$ . As  $M$  can take on the even values from 4 to 16, the 3-bit field can take on the values from 1 to 7. The  $L$  field is assigned the value of  $4*(\text{bit } 2) + 2*(\text{bit } 1) + (\text{bit } 0)$  and encodes the size of the length field used to store  $l(m)$ . The parameter  $L$  can take on the values from 2 to 8 (the value  $L=1$  is reserved). This value is encoded in the 3-bit field using the values from 1 to 7 by choosing the field value as  $L-1$  (the zero value is reserved).

If  $l(a)>0$  (as indicated by the Adata field) then one or more blocks of authentication data are added. These blocks contain  $l(a)$  and  $a$  encoded in a reversible manner. We first construct a string that encodes  $l(a)$ .

If  $0 < l(a) < 2^{16} - 2^8$  then the length field is encoded as two octets which contain the value  $l(a)$  in most-significant-octet first order.

If  $2^{16} \cdot 2^8 = <l(a) < 2^{32}$  then the length field is encoded as six octets consisting of the octets 0xff, 0xfe, and four octets encoding  $l(a)$  in most-significant-octet-first order.

If  $2^{32} = <l(a) < 2^{64}$  then the length field is encoded as ten octets consisting of the octets 0xff, 0xff, and eight octets encoding  $l(a)$  in most-significant-octet-first order.

This is summarized in the table below. All fields are interpreted in most-significant-octet-first order.

First two octets	Followed by	Comment
0x0000		Reserved
0x0001 ... 0xFEFF		For $0 < l(a) < 2^{16} \cdot 2^8$
0xFF00 ... 0xFFFD		Reserved
0xFFFE	Four octets $l(a)$	For $2^{16} \cdot 2^8 = < l(a) < 2^{32}$
0xFFFF	Eight octets $l(a)$	For $2^{32} = < l(a) < 2^{64}$

**Table A.5 – Length encoding for additional authentication data**

The blocks encoding  $a$  are formed by concatenating this string that encodes  $l(a)$  with  $a$  itself, and splitting the result into 16-octet blocks, padding the last block with zeroes if necessary. These blocks are appended to the first block  $B_0$ .

After the (optional) additional authentication blocks have been added, we add the message blocks. The message blocks are formed by splitting the message  $m$  into 16-octet blocks, padding the last block with zeroes if necessary. If the message  $m$  consists of the empty string, then no blocks are added in this step.

The result is a sequence of blocks  $B_0, B_1, \dots, B_n$ . The CBC-MAC is now computed by:

$$X_1 := E_k(B_0)$$

$$X_{i+1} := E_k(X_i \text{ XOR } B_i) \text{ for } i = 1, \dots, n$$

$$T := \text{first-}M\text{-octets}(X_{n+1})$$

where  $E_k()$  is the AES block cipher encryption function using key  $K$ , and  $T$  is the MAC value. Note that the last block  $B_n$  is exclusive-ORed with  $X_n$  and encrypted with the block cipher to give  $T$ .

### A.4.3 Encryption

To encrypt the message data we use AES CTR mode. We first define the key stream blocks by:

$$S_i := E_k(A_i) \quad \text{for } i = 0, 1, 2, \dots$$

The values  $A_i$  are formatted as shown in the table below.

<b>Octet #:</b>	0	1 ... 15- $L$	16- $L$ ... 15
<b>Contents:</b>	Flags	Nonce $N$	Counter $i$

**Table A.6 – Encryption blocks  $A_i$**

where  $i$  is encoded in most-significant-octet-first order. The **Flags** field is formatted as shown in the table below.

<b>Bit #:</b>	7	6	5	4	3	2	1	0
<b>Contents:</b>	Reserved	Reserved	0			L		

**Table A.7 – Encryption flags octet**

The Reserved bits shall be set to zero. Bit 6 corresponds to the Adata bit in the  $B_0$  block, but as this bit is not used here, it is reserved. Bits 3, 4, and 5 are set to 0. This ensures that all the  $A$  blocks are distinct from  $B_0$ , which has the non-zero encoding of  $M$  in this position. Bits 0, 1, and 2 contain  $L$ , using the same encoding as in  $B_0$ .

The message is encrypted by exclusive-ORing the octets of message  $m$  with the first  $l(m)$  octets of the concatenation of  $S_1, S_2, \dots$ . Note that  $S_0$  is not used to encrypt the message.

The authentication value  $U$  is computed by encrypting  $T$  with the key stream block  $S_0$  and truncating it to the desired length.

$$U := T \text{ XOR first-}M\text{-octets}(S_0)$$

#### A.4.4 Output

The final result  $c$  consists of the encrypted message  $m$ , followed by the encrypted authentication value  $U$ .

#### A.4.5 Decryption

To decrypt a message the following information is required:

- The 16-octet encryption key  $K$ .
- The  $(15-L)$ -octet nonce  $N$ .
- The additional authenticated data  $a$ .
- The encrypted and authenticated message  $c$ .

Decryption starts by re-computing the key stream to recover the message  $m$  and the MAC value  $T$ . The message and additional authentication data is then used to recompute the CBC-MAC value and check  $T$ .

If the  $T$  value is not correct, the receiver shall not reveal any information except for the fact that  $T$  is incorrect. In particular, the receiver shall not reveal the decrypted message, the value  $T$ , or any other information.

#### A.4.6 Restrictions

All implementations shall limit the total amount of data that is encrypted with a single key. The sender shall ensure that the total number of block cipher encryption operations in the CBC-MAC and encryption together shall not exceed  $2^{61}$ . This allows close to  $2^{64}$  octets to be encrypted and authenticated using CCM, which should be more than enough for most applications. Receivers that do not expect to decrypt the same message twice may also implement this limit. The recipient shall verify the CBC-MAC before releasing any information such as the plaintext. If the CBC-MAC verification fails, the receiver shall destroy all information, except for the fact that the CBC-MAC verification failed.

#### A.4.7 List of symbols

The table below provides a list of the symbols used in the preceding specification of AES CCM mode.

<b>Name</b>	<b>Description</b>	<b>Size</b>	<b>Comment</b>
$a$	Additional authenticated data	$l(a)$ octets	Use empty string if not desired
$A_t$	Counter block to generate key stream	16 octets	Contains block counter, nonce, and flags
$B_t$	Input block for CBC-MAC	16 octets	Together encode $N$ , $L$ , $M$ , $m$ , and $a$ uniquely
$c$	Ciphertext	$L(m) + M$ octets	Includes the encrypted MAC
$K$	Block cipher key	16 octets	Future expandable to 32 octets
$L$	Number of octets in length field	3 bits	Values 1 ... 8, encoded in 3 bits as $L-1$
$m$	Message to be encrypted and sent	$l(m)$ octets	Subject to $0 = l(m) < 2^{8L}$
$M$	Number of octets in authentication field	3 bits	Values 4,6,8, ..., 16. Encoded value is $(M-2)/2$
$N$	Nonce	$15-L$ octets	Nonce should never be repeated for same key.
$S_t$	Block of the encryption key stream	16 octets	Use $S_0, S_1, S_2, \dots$ to encrypt $m$ and $T$ .
$T$	Unencrypted authentication tag	$M$ octets	

$U$	Encrypted authentication tag	$M$ octets	Appended to the message after encryption
$X_t$	Intermediate value of CBC-MAC	16 octets	

## A.5 AES CBC-MAC Mode Operation

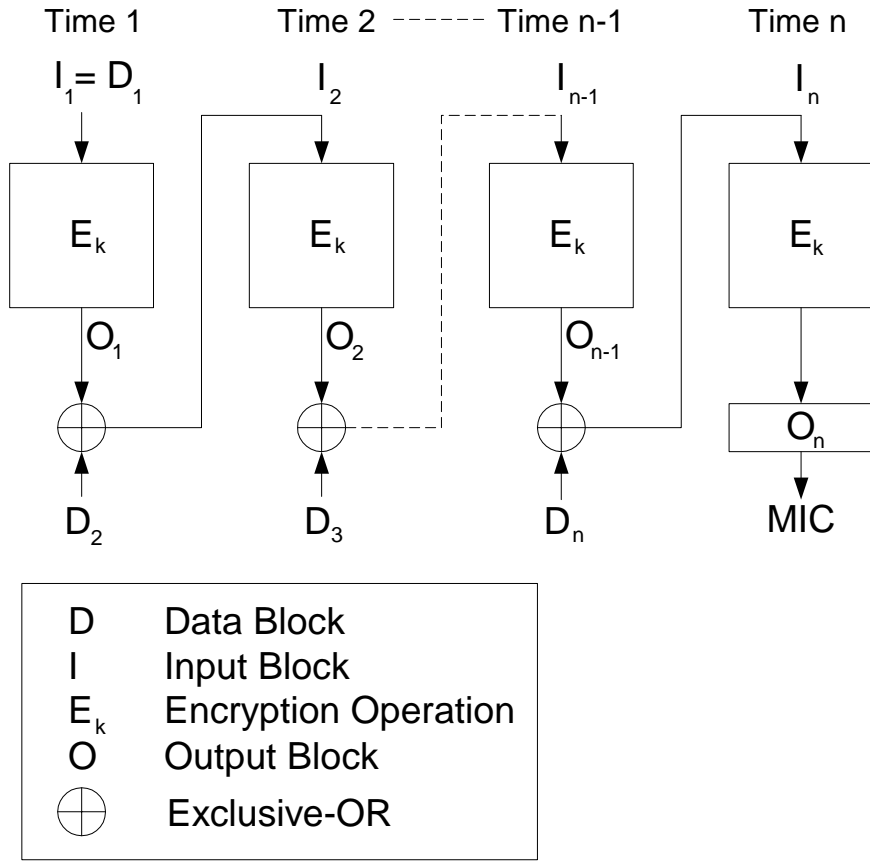
The CBC-MAC algorithm makes use of the underlying AES block cipher to provide data integrity on input data. The AES block cipher transforms (or encrypts) input vectors of the 128-bit block size to output vectors of the 128-bit block size using a 128-bit cryptographic key. Let  $D$  be any input vector and assume a key has been selected. The vector of length equal to the block size,  $O$ , which is the output of the AES block cipher when applied to  $D$ , using the enciphering operation, is represented as follows.

$$O = E_k(D)$$

The data (e.g. record, file, message, or program) to be authenticated is grouped into contiguous blocks:  $D_1, D_2, \dots, D_n$  each with length equal to the block size. If the number of data bits is not a multiple of the block size, then the final input block will be a partial block of data, left justified, with zeroes appended to form a full block. The calculation of the MIC (message integrity code) is given by the following equations.

$$\begin{aligned} O_1 &= E_k(D_1) \\ O_2 &= E_k(D_2 \text{ XOR } O_1) \\ O_3 &= E_k(D_3 \text{ XOR } O_2) \\ &\dots \\ O_n &= E_k(D_n \text{ XOR } O_{n-1}) \end{aligned}$$

The MIC is selected from  $O_n$ . Devices that implement CBC-MAC according to this standard will select all 128-bits of  $O_n$  as the MIC. A diagram showing the MIC generation technique for CBC-MAC mode is included below.



**Figure A.4 – AES CBC-MAC Mode**